GENERALITAT VALENCIANA
Conselleria d'Educació, Investigació, Cultura i Esport

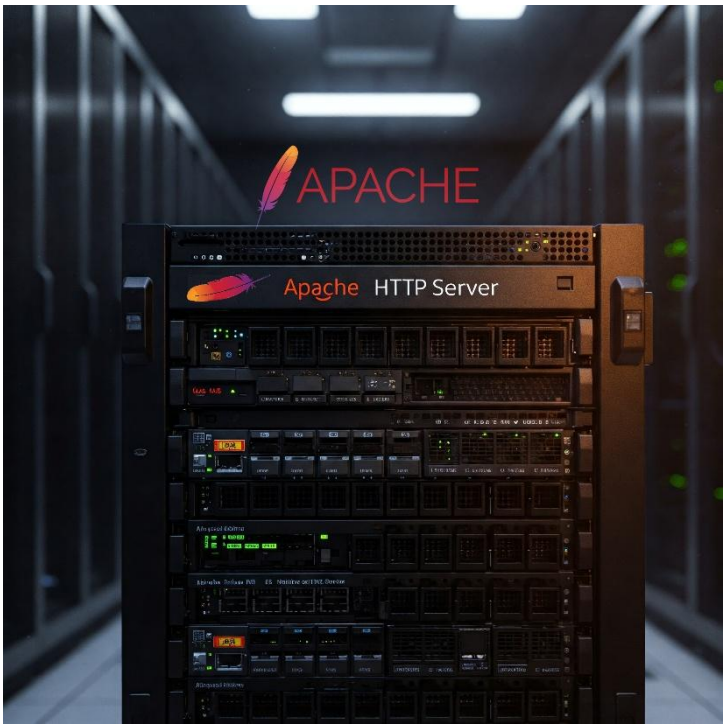
Institut Educació Secundària El Caminàs

Unió Europea
Fons Social Europeu
El FSE inverteix en el teu futur

Ciclo de Especialización

Ciberseguridad

Apache Hardening



Javier Alejandro Casas Arias

31/01/2026

Índice

Introducción

Esta memoria presenta los resultados de asegurar un servidor Apache desplegado en contenedores Docker, aplicando medidas de seguridad progresivas. Se implementaron políticas CSP, WAF con ModSecurity, reglas OWASP CRS, protección ante ataques DDoS, certificados SSL/TLS y buenas prácticas de hardening. Cada paso se verificó mediante pruebas prácticas y logs, demostrando la efectividad de las configuraciones en la protección del servidor frente a amenazas web comunes.

Desarrollo

En esta memoria de proyecto se muestran los principales resultados obtenidos al aplicar una serie de medidas de seguridad secuenciales en un servidor Apache, mediante contenedores de Docker.

Práctica 1: CSP

A continuación, se detalla la creación del contenedor **apache-server** a partir de la imagen base de esta actividad. El resto de las prácticas dependen de este paso inicial, ya que se ha diseñado un flujo de trabajo en cascada que permite heredar las configuraciones previas.

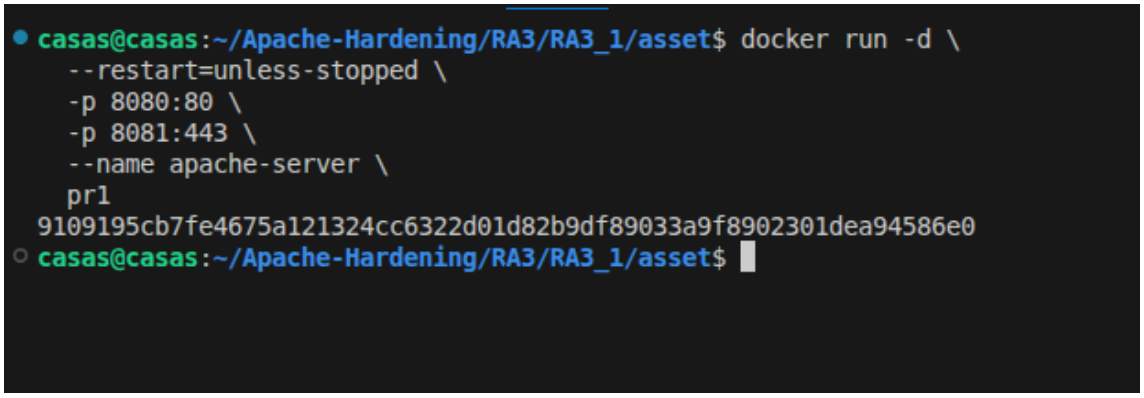


Imagen 1: Crear contendor apache-server

La captura muestra la verificación de las cabeceras de seguridad mediante el comando curl -I. Se confirma la correcta implementación de:

- Restricción de carga de recursos a fuentes autorizadas para mitigar ataques XSS.
- Aplicación efectiva de las políticas tanto en el puerto 8080 (HTTP) como en el 8081 (HTTPS).
- Forzado de conexiones seguras HTTPS.

```
casas@casas:~/Apache-Hardening/RA3/RA3_1$ curl -I http://localhost:8080/
HTTP/1.1 200 OK
Date: Fri, 23 Jan 2026 13:28:21 GMT
Server: Apache/2.4.58 (Ubuntu)
Strict-Transport-Security: max-age=31536000; includeSubDomains
Last-Modified: Fri, 23 Jan 2026 13:06:14 GMT
ETag: "e7-6490dd2764bd8"
Accept-Ranges: bytes
Content-Length: 231
Vary: Accept-Encoding
Content-Security-Policy: default-src 'self'; img-src *; media-src media1.com media2.com; script-src userscripts.example.com
Content-Type: text/html

casas@casas:~/Apache-Hardening/RA3/RA3_1$ curl -I https://localhost:8081/ --insecure
HTTP/1.1 200 OK
Date: Fri, 23 Jan 2026 13:28:48 GMT
Server: Apache/2.4.58 (Ubuntu)
Strict-Transport-Security: max-age=31536000; includeSubDomains
Last-Modified: Fri, 23 Jan 2026 13:06:14 GMT
ETag: "e7-6490dd2764bd8"
Accept-Ranges: bytes
Content-Length: 231
Vary: Accept-Encoding
Content-Security-Policy: default-src 'self'; img-src *; media-src media1.com media2.com; script-src userscripts.example.com
Content-Type: text/html

casas@casas:~/Apache-Hardening/RA3/RA3_1$
```

Imagen 2: Comprobar configuración de hardening

En la siguiente imagen se puede observar como se accede a nuestro sitio web mediante conexión segura con el protocolo HTTPS.

Imagen 3: Acceso a la web por HTTPS

Práctica 2. Web Application Firewall (WAF)

Creamos el contendor apache-waf partiendo de la imagen pr2.

```
casas@casas:~/Apache-Hardening/RA3/RA3_2/asset$ docker run -d \
  -p 8082:80 \
  -p 8083:443 \
  --name apache-waf \
  pr2
3e5cd56102e769a1856ec789388321352c8917a82d40c9da89a8b45d51f8e52e
casas@casas:~/Apache-Hardening/RA3/RA3_2/asset$
```

Imagen 4: Crear contenedor apache-waf

En la siguiente imagen se envía texto legítimo, donde la petición POST se procesa correctamente y el servidor responde sin bloquearla. Para ello nos dirigimos al navegador e ingresamos la siguiente ruta.

Imagen 5: Realizar una petición válida

En las siguientes imágenes se muestra que, al introducir código malicioso, Modsecurity detecta el patrón de ataque definido en sus reglas y lo bloquea.

Imagen 6: Introducir código malicioso

El servidor devuelve el error 403 Forbidden, permitiendo comprobar que el WAF está activo.

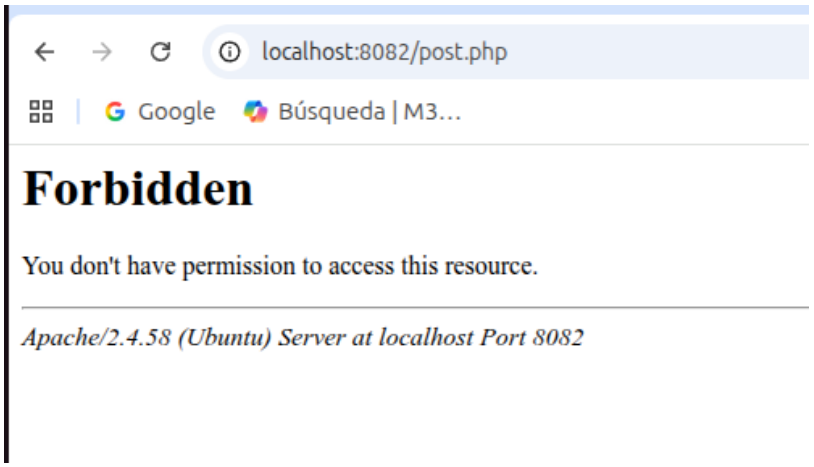


Imagen 7: Comprobación funcionamiento WAF (403 Forbidden)

Practica 3: Instalación reglas OWASP

El objetivo de esta actividad práctica es aumentar la seguridad del servidor Apache con el cual se ha trabajado hasta el momento. Teniendo esto en cuenta se ha instalado y configurado ModSecurity con las reglas OWASP CRS en Apache para proteger el servidor frente a ataques web comunes.

Para crear el contenedor **apache-owasp** se debe utilizar el siguiente comando.

```
casas@casas:~/Apache-Hardening/RA3/RA3_3/asset$ docker run -d -p 8084:80 -p 8085:443 --name apache-owasp pr3
7a118cd08e4b48941c55334ed7d3b9bad20a5e8560d83ba94f732c6e661a60d2
casas@casas:~/Apache-Hardening/RA3/RA3_3/asset$
```

Imagen 8: Crear contenedor apache-owasp

El siguiente comando envía una petición HTTP al servidor Apache con un parámetro sospechoso (testparam=test) usando curl y sirve para comprobar que ModSecurity y las reglas OWASP están funcionando, ya que la respuesta 403 Forbidden confirma que la petición ha sido detectada y bloqueada por una regla de seguridad.

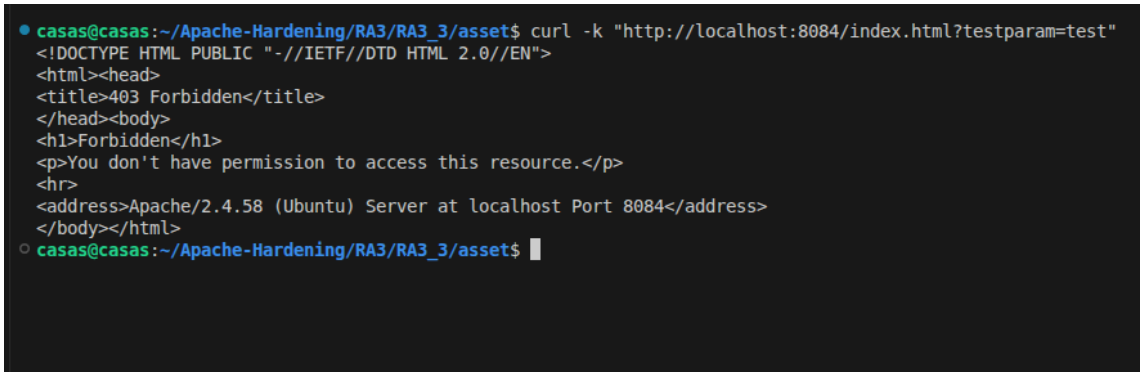


Imagen 9. Comprobar funcionamiento de ModSecurity

Los siguientes logs muestran que Modsecurity está funcionando correctamente y bloqueando peticiones maliciosas. Los logs muestran que una solicitud al recurso /index.html fue denegada con código 403 porque el parámetro testparam contenía el valor test, lo que hizo que la regla personalizada con ID 999999 definida en custom-rules.conf se activara. También se muestra como indica en el banner de los logs nuestro mensaje personalizado: **ModSecurity lo ha Bloqueado.**

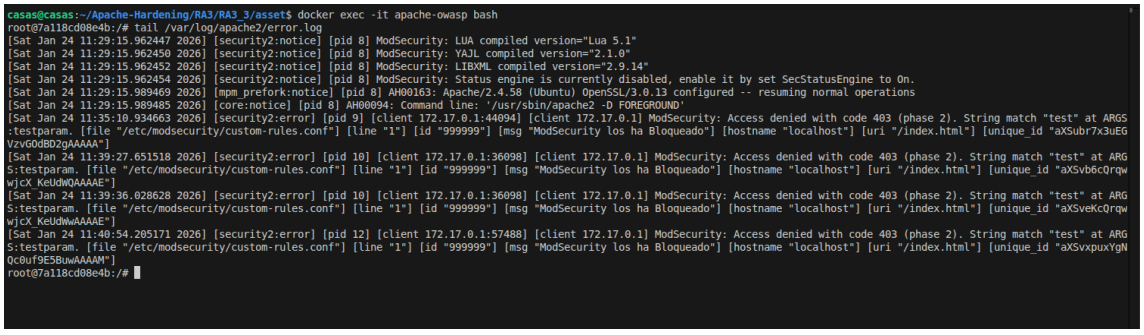


Imagen 10. Comprobación de logs

Práctica 4: Evitar ataques DDoS

El objetivo de esta actividad es instalar y configurar el módulo mod_evasive en Apache para prevenir ataques de denegación de servicio (DoS), comprobar su funcionamiento mediante Apache Bench y crear una imagen Docker que deje esta protección instalada, configurada y verificable.

En la siguiente imagen se muestra el comando utilizado para crear el contenedor **apache-ddos.**

```
casas@casas:~/Apache-Hardening/RA3/RA3_4/asset$ ls
Dockerfile  evasive.conf
casas@casas:~/Apache-Hardening/RA3/RA3_4/asset$ docker run -d -p 8086:80 -p 8087:443 --name apache-ddos pr4
cde6b909bafe3dd99f8f9c40b424981b019b4707c1c125122e9675ddb8f483d2
casas@casas:~/Apache-Hardening/RA3/RA3_4/asset$
```

Imagen 11: Crear contenedor apache-ddos

Con el siguiente comando se ha simulado una carga intensa de tráfico contra el servidor web y se comprueba como responde Apache cuando recibe muchas peticiones simultáneas.

Concretamente, ab -n 3000 -c 50 http://localhost:8086/ envía 3000 peticiones HTTP en total (-n 3000) con un nivel de 50 conexiones concurrentes (-c 50) al servidor.

```
casas@casas:~/Apache-Hardening/RA3/RA3_4/asset$ ab -n 3000 -c 50 http://localhost:8086/
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 300 requests
Completed 600 requests
Completed 900 requests
Completed 1200 requests
Completed 1500 requests
Completed 1800 requests
Completed 2100 requests
Completed 2400 requests
Completed 2700 requests
Completed 3000 requests
Finished 3000 requests


Server Software:        Apache/2.4.58
Server Hostname:        localhost
Server Port:            8086

Document Path:          /
Document Length:        10671 bytes

Concurrency Level:      50
Time taken for tests:   0.263 seconds
Complete requests:      3000
Failed requests:        2985
   (Connect: 0, Receive: 0, Length: 2985, Exceptions: 0)
Non-2xx responses:      2985
Total transferred:      1527210 bytes
HTML transferred:       983925 bytes
Requests per second:    11422.87 [#/sec] (mean)
Time per request:       4.377 [ms] (mean)
Time per request:       0.088 [ms] (mean, across all concurrent requests)
Transfer rate:          5678.75 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.2      0       2
Processing:     1    4   3.0      4      28
Waiting:        1    4   3.0      4      28
Total:          2    4   3.1      4      30

Percentage of the requests served within a certain time (ms)
  50%      4
  66%      4
  75%      4
  80%      4
  90%      5
  95%      5
  98%      5
  99%     27
 100%     30 (longest request)
casas@casas:~/Apache-Hardening/RA3/RA3_4/asset$
```

Imagen 12. Simular ataque de denegación de servicio

El siguiente comando entra en el contenedor Docker y consulta los logs de error de Apache filtrando los eventos de mod_evasive, para comprobar si el módulo está bloqueando peticiones.

Lo que se observa en la salida indica que las peticiones procedentes de la IP 172.17.0.1 están siendo denegadas, lo que confirma que mod_evasive ha detectado un comportamiento abusivo, bloqueando las conexiones correctamente.

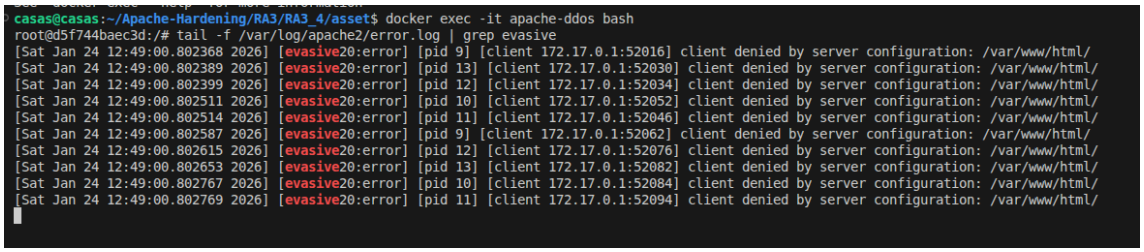


Imagen 13. Revisar logs de error en Apache

Práctica 5: Certificados

El objetivo de esta práctica es instalar y configurar un nuevo certificado digital SSL/TLS en Apache para cifrar las comunicaciones entre el cliente y el servidor. De esta manera, se puede verificar el funcionamiento de HTTPS mediante un certificado auto firmado, comprender sus limitaciones frente a certificados emitidos por una autoridad de confianza, y configurar la redirección automática de HTTP a HTTPS.

Esta comprobación realizada desde el contendor **certificados** demuestra que el servidor Apache dentro del contenedor está correctamente configurado para servir contenido HTTPS usando el dominio configurado, con un certificado SSL funcional y un VirtualHost bien definido, validando la configuración interna del servidor.

```
root@bb037c3b84f9:/# curl -k https://www.midominioseguro.com
<!DOCTYPE html>
<html>
        <head>
                <title>Sitio Web Apache</title>
        </head>
        <body>

                <h1>Sitio web de apache pps</h1>
                <h2>Javier Alejandro Casas</h2>
                <h3>Sitio para la seguridad CSP y HSTS</h3>
        </body>
</html>
root@bb037c3b84f9:/#
```

Imagen 14: Realizar comprobación del certificado desde el contenedor

También se ha realizado una comprobación desde el navegador con el objetivo de acceder a nuestro sitio web de manera segura.

En la siguiente imagen se puede observar como se ha podido acceder al sitio web:

Imagen 15: Comprobar funcionamiento del certificado desde el navegador.

Si accedemos a detalles generales del certificado, podemos observar como nuestro certificado está correctamente asociado al dominio **www.midominioseguro.com**.

Imagen 16: Descripción del certificado

A continuación, se muestra el comando utilizado para comprobar que se fuerza la redirección de HTTP a HTTPS cuando se accede a nuestro sitio web.

```
casas@casas:~/Apache-Hardening/RA3/RA3_2/asset$ curl -I http://www.midominioseguro.com:8090
HTTP/1.1 301 Moved Permanently
Date: Mon, 26 Jan 2026 20:28:42 GMT
Server: Apache/2.4.58 (Ubuntu)
Strict-Transport-Security: max-age=31536000; includeSubDomains
Location: https://www.midominioseguro.com/
Content-Type: text/html; charset=iso-8859-1

casas@casas:~/Apache-Hardening/RA3/RA3_2/asset$
```

Imagen 17. Comprobar redirección de HTTP a HTTPS

Práctica 6: Mejores prácticas de endurecimiento de Apache

El objetivo de esta actividad es aplicar las mejores prácticas de endurecimiento del servidor Apache en el que hemos ido trabajando hasta el momento.

Este comando sirve para validar de forma rápida y limpia que Apache está bien configurado y protegido a nivel de cabeceras HTTP.

```
casas@casas:~/Apache-Hardening/RA3/RA3_3/asset$ curl -I -k https://localhost:8093/
HTTP/1.1 200 OK
Date: Mon, 26 Jan 2026 21:48:39 GMT
Server: Apache
Strict-Transport-Security: max-age=31536000; includeSubDomains
X-Frame-Options: SAMEORIGIN
Last-Modified: Fri, 23 Jan 2026 12:05:50 GMT
ETag: "29af-6490cfa717831"
Accept-Ranges: bytes
Content-Length: 10671
Vary: Accept-Encoding
Content-Security-Policy: default-src 'self'; img-src *; media-src media1.com media2.com; script-src userscripts.example.com
X-XSS-Protection: 1; mode=block
Content-Type: text/html

casas@casas:~/Apache-Hardening/RA3/RA3_3/asset$
```

Imagen 18: Comprobar configuración y protección de cabeceras

El siguiente comando realiza una comprobación de seguridad para verificar si el servidor Web tiene protección contra ataques **Cross-Site-Scripting (XSS)**

```
casas@casas:~/Apache-Hardening/RA3/RA3_3/asset$ curl -v -k "https://localhost:8093/?test=<script>alert(1)</script>"
* Host localhost:8093 was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
*   Trying [::1]:8093...
* Connected to localhost (::1) port 8093
* ALPN: curl offers h2,http/1.1
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384 / X25519 / RSASSA-PSS
* ALPN: server accepted http/1.1
* Server certificate:
*  subject: C=ES; ST=Castellon; L=Castellon de la Plana; O=Ciberseguridad; OU=Sistemas; CN=www.midominioseguro.com
*  start date: Jan 28 21:27:04 2026 GMT
*  expire date: Jan 28 21:27:04 2027 GMT
*  issuer: C=ES; ST=Castellon; L=Castellon de la Plana; O=Ciberseguridad; OU=Sistemas; CN=www.midominioseguro.com
*  SSL certificate verify result: self-signed certificate (18), continuing anyway.
*   Certificate level 0: Public key type RSA (2048/112 Bits/secBits), signed using sha256WithRSAEncryption
* using HTTP/1.x
> GET /?test=<script>alert(1)</script> HTTP/1.1
> Host: localhost:8093
> User-Agent: curl/8.5.0
> Accept: */*
>
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* old SSL session ID is stale, removing
< HTTP/1.1 403 Forbidden
< Date: Sat, 31 Jan 2026 12:02:21 GMT
< Server: Apache
< Strict-Transport-Security: max-age=31536000; includeSubDomains
< X-Frame-Options: SAMEORIGIN
< Content-Length: 199
< Content-Type: text/html; charset=iso-8859-1
<
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
</body></html>
* Connection #0 to host localhost left intact
casas@casas:~/Apache-Hardening/RA3/RA3_3/asset$
```

Imagen 19: Comprobar protección ante ataques XSS

El siguiente comando intenta usar una función de diagnóstico potencialmente peligrosa y el servidor la rechazó con un error 405. Esto confirma que el servidor Web está protegido contra ataques de Cross-Site Tracing (XST). Además, se puede observar que no muestra la versión de Apache, impidiendo que un

atacante pueda trazar un plan de ataque identificando vulnerabilidades conocidas de determinadas versiones.

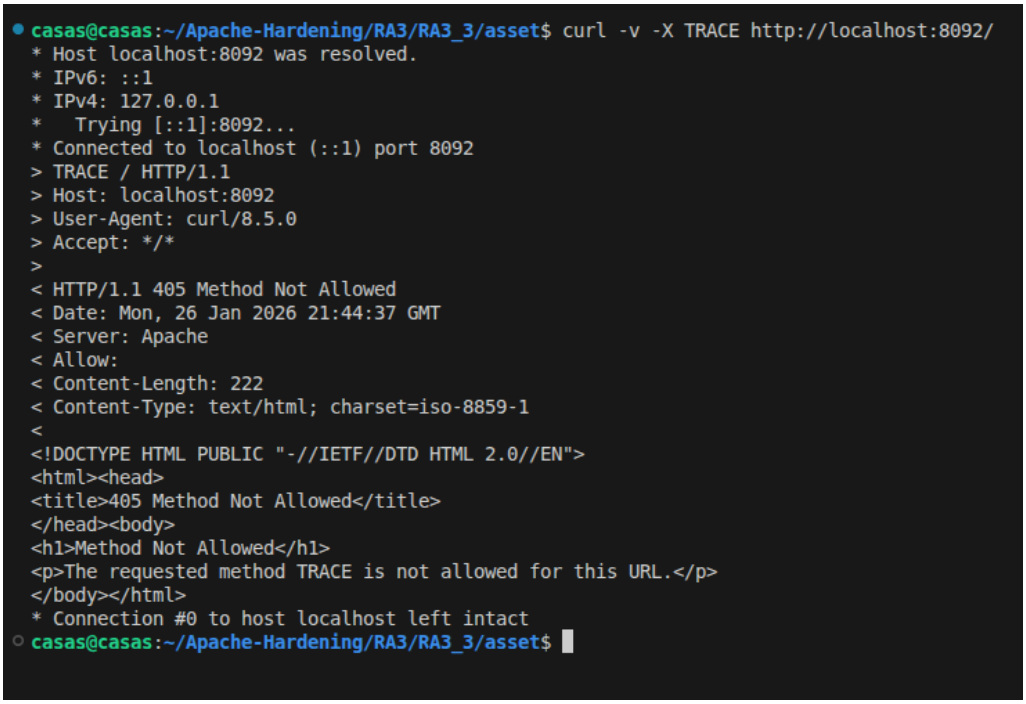


Imagen 20: Realizar comprobación de diagnóstico sobre el servidor web

Bibliografía

Pérez Garrac, C. *Hardening del servidor*. Ciberseguridad-PePS.

https://psegarrac.github.io/Ciberseguridad-PePS/tema3/seguridad/web/2021/03/01/Hardening-Servidor.html

Pérez Garrac, C. *Certificado digital (P1-SSL)*. Ciberseguridad-PePS.

https://psegarrac.github.io/Ciberseguridad-PePS/tema1/practicas/2020/11/08/P1-SSL.html

Geekflare. (s. f.). *Apache Web Server Hardening and Security Guide*.

https://geekflare.com/cybersecurity/apache-web-server-hardening-security/

Conclusiones

Las medidas aplicadas reforzaron la seguridad del servidor Apache de manera comprobable: CSP y WAF bloquearon ataques XSS y peticiones maliciosas; OWASP CRS aumentó la protección ante vulnerabilidades conocidas; mod_evasive mitigó ataques DDoS; y los certificados SSL/TLS garantizaron comunicaciones cifradas. El hardening final aseguró cabeceras y ocultó información sensible, logrando un servidor web más seguro, confiable y resistente ante ataques.